



DyRT

Dynamic Response Textures
for Real Time Deformation
Simulation with Graphics Hardware

Doug L. James & Dinesh K. Pai
University of British Columbia

Motivation for DyRT



What is DyRT?

- Dynamic Response Texture (DyRT)
- Dynamic physically based deformations
- Respond to rigid bone motion
 - Character animation
- Synthesized in shader hardware
- DyRT is cheap!



Related Work

Related Work: Real Time Deformation

- Extensive related work [Barri84, TeraFleischer88, Chadwick89, Courter89, PentlandWilliams89, WitkinWelch90, MetaxasTerz92, BaraffWitkin92, WilhelmsVanGelder97, ...]
- Multiires adaptive approaches [ZhuangCanny00, Wu01, Debeaux01, Pieakone01, ...]
- Precomputed data-driven deformation
- Fast Green's function methods for linear elastostatics [Cotin99, JamesPai99-02]
 - Multizone [JamesPai ICRA02]
- EigenSkin, w/ Paul Kry, SCA02

Related Work: Modal Simulation

- Modal analysis is a standard tool
- Pentland & Williams, *Good Vibrations*, SIGGRAPH 89
- Interactive precomputed modal models:
 - Stochastic dynamics [Stam97]
 - Contact sounds [DoelPai96, DoelKryPai01, O'Brien02]
 - Force-feedback applications [Basdogan01]

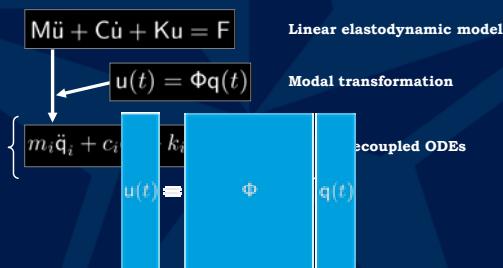
Our Contributions

- Dynamic physically based deformations synthesized almost entirely in graphics hardware
- Negligible CPU cost
- Driven by any rigid motion, e.g., bone-based animation

Talk Overview

- Background: Modal Analysis
- Exciting modes with rigid motion
- Mapping on to graphics hardware
- Recipe for DyRT
- Results

Background: Modal Analysis



Dynamic Integration

- Simple harmonic oscillator
 $m_i\ddot{q}_i + c_i\dot{q}_i + k_i q_i = Q_i$
- Use small IIR digital filter: (mode i , time k)
 $q_i^{(k)} = \alpha_i q_i^{(k-1)} + \beta_i \dot{q}_i^{(k-1)} + \gamma_i Q_i^{(k-1)}$
- Precompute IIR filters...

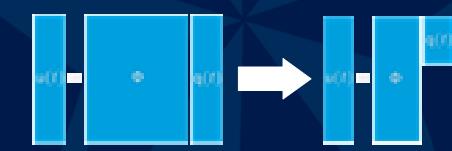
Background: Modal Analysis

Interactive Video Tutorial



Background: Modal Truncation

- Can often truncate higher modes
 - Lowest modes are dominant
 - Higher modes heavily damped & die out fast





Exciting Modes with Rigid Motions

- Accelerations produce inertial forces

Exciting Modes with Rigid Motions

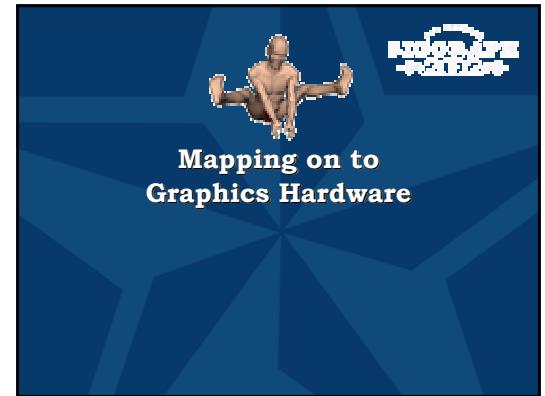
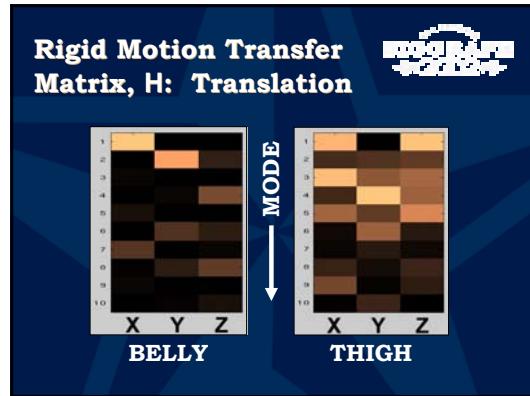
- Rigid motion transfer matrix
 - m -by-6 matrix of precomputed volume integrals
- Spatial velocity, $\dot{\psi} = \begin{bmatrix} \omega \\ v \end{bmatrix}$

$$(\psi^{(k)} - \psi^{(k-1)}) \xrightarrow{H} \text{force: } Q^{(k)}$$

Exciting Modes with Rigid Motions

- Matrix picture of modal forces...

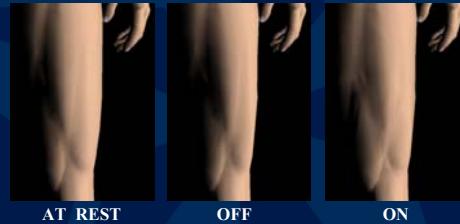
$$Q^{(k)} = H (\psi^{(k)} - \psi^{(k-1)})$$

$$m \left[\begin{array}{c|c} Q & \\ \hline & \end{array} \right] = \left[\begin{array}{c|c} \begin{matrix} \textcolor{green}{\omega} \\ \textcolor{blue}{\omega} \\ H \end{matrix} & \begin{matrix} \textcolor{red}{v} \\ \textcolor{green}{v} \\ \textcolor{blue}{v} \end{matrix} \end{array} \right] \left[\begin{array}{c|c} & 6 \\ \hline & \end{array} \right]$$


Programmable Graphics Hardware

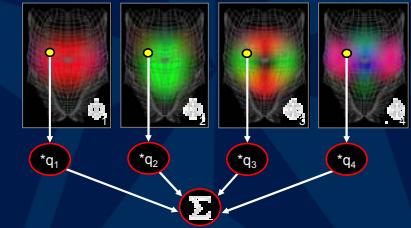
- Vertex/pixel stream processing
- Lindholm, Kilgard & Moreton, *A User-Programmable Vertex Engine*, SIGGRAPH 2001
- DyRT vertex programs
- Modes give per-vertex displacements
- No connectivity available for normals
⇒ But can easily compute linear correction

Effect of linearized normal correction...



DyRT Vertex Program

- Per-vertex data: Φ_{im} , N_{im}
- E.g., 4-mode DyRT displacement



DyRT Vertex Program

```
# Load vertex  $p_i$  into R1 and add 5 modal corrections:
MOV R1, v[0POS];                                # R1 =  $p_i$ 
MAD R1, c[DyRT 1].xxxw, v[5], R1;      # R1 +=  $q_1 \Phi_{i1}$ 
MAD R1, c[DyRT 1].yyvw, v[6], R1;      # R1 +=  $q_2 \Phi_{i2}$ 
MAD R1, c[DyRT 1].zzzw, v[7], R1;      # R1 +=  $q_3 \Phi_{i3}$ 
MAD R1, c[DyRT+1].xxxw, v[8], R1;      # R1 +=  $q_4 \Phi_{i4}$ 
MAD R1, c[DyRT+1].yyvw, v[9], R1;      # R1 +=  $q_5 \Phi_{i5}$ 

# Load normal  $n_i$  into R2 and add 5 modal corrections:
MOV R2, v[NRML];                                # R2 =  $n_i$ 
MAD R2, c[DyRT 1].xxxw, v[10], R2;     # R2 +=  $q_1 N_{i1}$ 
MAD R2, c[DyRT 1].yyvw, v[11], R2;     # R2 +=  $q_2 N_{i2}$ 
MAD R2, c[DyRT 1].zzzw, v[12], R2;     # R2 +=  $q_3 N_{i3}$ 
MAD R2, c[DyRT+1].xxxw, v[13], R2;     # R2 +=  $q_4 N_{i4}$ 
MAD R2, c[DyRT+1].yyvw, v[14], R2;     # R2 +=  $q_5 N_{i5}$ 
```



Recipe for DyRT

- Precomputation
- Runtime Synthesis

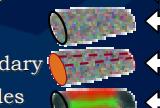
DyRT Process: Precomputation

- Define solid model
 - Anatomically based modeling

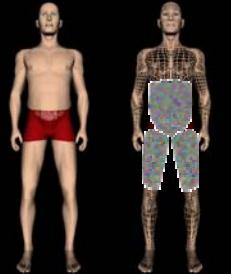


- Finite element modal analysis:

- Mesh volume
- Constrain non-exposed boundary
- Compute m leading eigenmodes



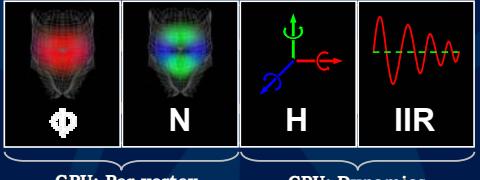
Precomputing DyRT-Man



- Leg model
 - 10k tetrahedra
- Belly model
 - 30k tetrahedra
- Time: a few minutes
- Interpolate modes back onto character surface
- Reusable

DyRT Process: Precomputation

- Construct m -mode DyRT object



GPU: Per-vertex deformation data CPU: Dynamics

DyRT Process: Runtime Synthesis

- For each DyRT model...
 - Compute modal forcing
 - Integrate dynamics $\xrightarrow{\text{IIR}}$ $q^{(k)}$
 - Bind DyRT vertex program
 - Set program constants ($q^{(k)}$, ...)
 - Draw model, e.g., call display list

Results



Results: DyRT-Man!



Results: Laparoscopic surgical simulation

- DyRT applied to hanging fatty tissue
- Driven by semi-rigid coupling with scene



Summary & Conclusion

- Real time dynamic deformations
- Responds to rigid motion
 - Ideal for character animation
- Synthesized on GPU
- Negligible cost to main CPU
- Future work...

Real DyRT!



Acknowledgements

- Software
 - House of Moves
 - Curious Labs Poser
 - “GL4Java”
 - FEM software (TETGEN, CalculiX)
- Funding
 - Precarn (Inst. Robotics & Intelligent Systems)

